SCIENTIFIC REPORT NO. 3

AUTOMATIC QUESTION-ANSWERING OF ENGLISH-LIKE QUESTIONS ABOUT SIMPLE DIAGRAMS

by

Manfred Kochen

University of Michigan
Consultant to RCA Laboratories
Princeton, New Jersey 08540

Prepared for the Air Force Office of Scientific Research of the Office of Aerospace Research under Contract No. AF49(638)-1184.

PREFACE

This work was done during 1965 at RCA Laboratories in Princeton, N. J. It is part of our continuing research in the area of question-answering processes and their relationship to more general problems in machine problem solving. An RCA Laboratories report covering most of the material described here was issued in November 1965 under the title "Translation of English-like Queries into Efficient Computer Search Programs for Question-Answering." After several refinements and revisions the initial report has evolved into its present form, and it is being finalized for publication. Since the material presented here is relevant to much ongoing research, we are simultaneously issuing it as a scientific report in order to speed up its availability to the technical community.

Saul Amarel
Princeton, N. J.
May 1968

ABSTRACT

This paper presents a technique for translating certain English-like questions into procedures for answering them in order to explore how large a class of basic question types can be so processed. The English-like questions all pertain to simple diagrams built of elementary figures with relations like "above" and "larger than." The input to the program into which the algorithm presented here could be implemented are questions such as "Is it true that in Fig. 1 each triangle is above a circle," and may include terms like "how," "when," "what" in an interesting variety of interrogative sentence types. The output of the program is a flow diagram for another program to answer the question by inference and search of a structured data base in which representations of diagrams are stored.

The English-like source language of questions that the algorithm can process, though restricted and fixed in syntax and domain of discourse, has a potentially wide scope in that it includes some of the fundamental question types.

TABLE OF CONTENTS

I. INTRODUCTION

When Turing (38) proposed a test for distinguishing between the verbal behavior of a person alleged to betoken thinking and corresponding behavior on the part of a machine, a challenge presented itself to computer scientists. How can a computer be programmed to answer questions which resemble more and more the questions we ordinarily expect that only people can understand? During the past decade several dozen question-answering programs have been written, e.g., (16), (28), (29), (15), (9), (14), (33). Concurrently with and independently of the work reported in this paper (completed in 1965), a number of related studies appeared (10), (39), (35), (37).

A recent critical review (18) of this literature pointed out some major gaps in the theoretical underpinnings. This review concludes that "the only hope for success in the near future is in well-structured data-base systems, having a special internal structure appropriate to a specific field, a reliable technical language, and a competent inference mechanism."

Presented here is an algorithm for processing English-like questions. "Processing" as used above means: (a) parsing the question to analyze its syntactic structure; (b) building up, as a by-product of parsing, and simultaneously, the flow-diagram for a computer program; (c) the program if run, would search a structured data base for the answer to the question.

A typical question is: "Is it true that in Fig. 1, each circle which is inside a triangle is above a rectangle." The domain of discourse to which all questions and the data base is restricted consists of single

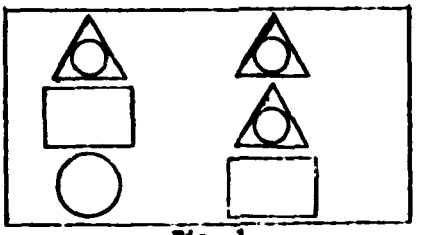


Fig. 1

diagrams composed of geometric figures arranged into various patterns like that shown in Fig. 1. A data base of such figures,* each represented for storage by descriptive forms like H (3, V (4,5), 6)) for the following figure,

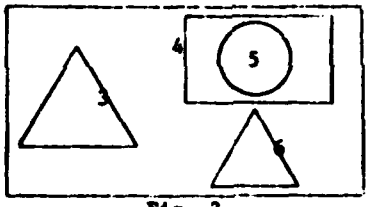


Fig. 2

for example, is well structured. The questions can be posed in terms of a minimum vocabulary including the predicates: H(a,b) = "a is to the left of b". V(a,b) = "a is above b" and I(a,b) = "a encloses b", with a and b being the names of individual elementary figures. Our aim, however, is to use the maximum, not the minimum, vocabulary for basic concepts that are common to many ordinary questions, and to explore how large a class of question types the algorithm can process.

The rules of inference used here are those of a classical applied predicate calculus, because nothing more sophisticated (e.g., modal logic) is required for the purpose on hand.

*This seems to have been used first by Minsky, and subsequently in (30), (20), (21), (29), (1), (27), (32).

While neither the class of problems studied here (39), (20), nor the methods of syntactic analysis and the accompanying translation techniques (35), (1), (2), (6), (7), (10), (34), (19), nor the use of rules of inference (5), (9), (12), (28) are novel, their combination is, and may suggest many interesting extensions.

II. THE SOURCE LANGUAGE: VOCABULARIES AND FIRST RULES OF FORMATION

To specify, as a formal language*, the class of questions which the algorithm processes is to specify the quintuple: $\{V_T, V_N, Q, R, T\}$. Here:

V_T is the set of words and phrases - the vocabulary - with which the questions are composed.

V_N is the set of "nonterminal" auxiliary symbols in terms of which the formation rules for generating the questions are expressed.

Q is a special element of V_N, which may be thought of as labeling the class of well-formed questions.

R is the collection of formation rules, which look like

$$Q \longrightarrow (QTRUE)\,(SENT) \quad \text{or} \quad QTRUE \longrightarrow \text{Is it true that.}$$

Such rules tell us that, from the symbol at the left of the arrow we may produce, generate, form, the "string" of symbols at the right. All words in capitals, which may be enclosed in parentheses, like Q, QTRUE, SENT are elements of V_N; each is thought of as a single symbol. We can form strings by concatenating - placing next to each other - such symbols, being sure to preserve the order in which they appear. In a rule of the second type, a phrase which is not all capitals appears to the right of the arrow. It is an element of V_T.

T is the collection of transformation rules, to be explained later. Beginning with the symbol Q, the rules R are applied until only elements of V_T appear in the resulting string; i.e., until all non-terminal symbols

*This method, or variants thereof, has been used in combinatorial linguistics (19), in syntax-directed compilers (17), in mechanical translation and language processing (4) (22) to mention but a few of the applications of this formal approach to linguistic systems first introduced in this way by logicians (6) (though some of the ideas stem from the 17th Century) and further developed by linguists (8).

like QTRUE, SENT etc., have been replaced by terminal vocabulary elements according to rules that allow this. The set of all questions that may be formed this way from the grammar $G = \{V_T, V_N, Q, R, T\}$ is called a formal language, L_G.

The reverse process of generating all the sentences of L_G from the starting symbol Q is called parsing. Beginning with a string of elements in V_T - a candidate question - we seek rules in R or T and try to apply them in a suitable order so that the given sentence could have somehow been generated from Q.

The grammar G for the source language we present next is restricted in comparison with a grammar for English or even for other source languages of question-answering algorithms, with regard to its syntax. Our aim, however, is to introduce elements of V_T which represents concepts that are fundamental to the content of a great variety of questions. That is, though the class of questions in L_G are syntactically all very similar, they can vary greatly in their logical content.

In Tables I and II we present the vocabularies V_T and V_N. The rules are numbered F 1 to F 11 and each is subdivided, as F 4.1, F 4.2, etc. Next to each vocabulary element is indicated the rule in which it is used. Note rules labeled T 2.15, etc., appear also. These are transformation rules described later.

We proceed next to R, the set of formation rules.

(F 10.4) $\quad Q \longrightarrow$ (QTRUE)(SENT)

The name or label of the rule by which we refer to it is written in parentheses at the left. We cannot, of course, exemplify the use of this rule

TABLE I

V_T

Element (Word or Phrase)	Rule Using It
above	F 3.1
after	F 1.21
and	F 1.8
an	F 1.5
a	T 2.15
before	F 1.22
below	F 3.2
changes occur	F 1.17
circle	T 2.1
circular	F 3.13
darker than	F 3.7
decreased	T 2.10
decreases	T 2.8, 2.6
decrease	T 2.9, 2.14, 2.12
decreasing	T 2.13, 2.11
did	T 2.12
do	F 1.24
each	F 1.6
enclosing	F 3.5, TR 3.5
figure	F 1.3
from where	F 1.18
how	F 1.15
if	F 1.10
increased	T 2.10
increases	T 2.8, 2.5
increase	T 2.9, 2.14, 2.12
increasing	T 2.13, 2.11
inside	F 3.6
in	F 1.2
is it false that	T 2.3
is it true that	F 1.1
is there	F 2.4
is	F 1.4
larger than	F 3.9
lighter than	F 3.8
moved	T 2.13
moves	T 2.4, 2.7
move	T 2.9, 2.14, 2.12
moving	T 2.11, 2.13
not	F 1.7
no	T 2.2
object	F 8.1
or	F 1.9
rectangle	T 2.1
rectangular	F 3.12
related to	F 1.16
smaller than	F 3.10
then	F 1.11
this sentence is uttered	F 1.25
to the left of	F 3.3
to the right of	F 3.4
to where	F 1.19
triangle	T 2.1
triangular	F 3.11
what	F 1.13
when	F 1.20
where	F 1.14
which	F 1.12
while	F 1.23
will	T 2.9

TABLE II

V_N

Element (Symbol)	Mnemonic Aid to interpretation	Rule Using It
AFT	after	F 10.17,10.19
AND	and	F 4.5,10.5
ANT	antecedent	F 5.2,5.3,6.1,6.2
BEF	before	F 10.17,10.20
CHOC	changes occur	F 10.14
CL	clause	F 6.2
CONS	consequent	F 4.3,4.5,4.6,7.1, 9,10.1,14.2 T 1.2,1.3
DO	do	F 10.17
FIG	figure	F 5.1
FRWH	from where	F 11.1,11.3
IF	if	F 10.8,T 1.2
IMP	implication	F 10.9,10.10
IN	in	F 5.1
IS	is	F 10.1,7.1,9,10.17 10.13
NI	name of an individual	F 4.1,5.1,10.1, 10.12,10.3
NOT	not	F 4.3,10.7
OR	or	F 4.6,10.6
POSPR2	2-place positional predicate	F 3.1, F 3.2
POST	post-fix	F 10.15
PRE	pre-fix	F 10.14
QUAL	qualifier	F 10.2-4,11.3,11.4 T 1.1,1.2,1.3,2.4
QUANT	quantifier	F 4.2, T 1.2
QUA	universal quantifier	F 5.3,10.10
QUE	exist partial quantifier	F 5.2,10.9, T 1.3
QTRUE	is it true that	F 10.2,10.4,10.3, T 1.1
SENT	sentence	F 10.1,10.21 T 1.1 F 11.1-4, T 1.1
SHPR1	1-place sharp predicate	F 4.4
STDSENT	"this sentence is uttered, standard sentence	F 10.18-21
THEN	then	F 10.8
TOWH	to where	F 11.2,11.4
VAR	variable	F 6.1, T 1.3
WHA	what	F 10.12, T 1.3
WHI	while	F 10.18,10.21
WHN	when	F 10.16,10.17,10.1
W	which	F 7.1 T 1.3

until we have introduced enough rules to define (specify what we can substitute for), QTRUE and SENT. The symbol SENT is a mnemonic label for all well-formed sentences. For completeness, however, we introduce two variants.

(F 10.2) Q ⟶ (QTRUE)(QUAL)(SENT)

(F 10.3) Q ⟶ (QUAL)(QTRUE)(SENT)

(F 5.1) QUAL⟶(IN)(FIG)(NI); QUAL is mnemonic for "qualifying clause"

The next few rules, all labeled (F 1.X), relate to terminal vocabulary words and phrases. We introduce only a partial list at this point, so that we can illustrate their use and justify their selection.

(F 1.1) QTRUE ⟶ Is it true that

(F 1.2) IN ⟶ in

(F 1.3) FIG ⟶ figure

(F 1.4) IS ⟶ is

Rules labelled (F 2.X)denote all the individual constants: the underlined numbers labelling unique, specific geometric object like circles, rectangles of which the patterns are built. No two circles, for example, are given the same label, and all the figures which constitute patterns enclosed in a rectangular frame, are also labelled: Figure 1, etc.

(F 2.1) NI ⟶ i i = 1, 2, 3 ---.

we used underlined numbers to label or refer to our figures and building blocks or primitive objects.

The next set of rules, like (F 1.X) and (F 2.X), also point to terminal words and phrases, but only to those indicating relations, or predicates,

with which sentences - the assertive parts of each question - are formed. The first is:

(F 3.1) POSPR2 ⟶ above. (POSPR2 is mnemonic for a 2-place position predicate)

We can now illustrate the use of the rules presented so far. Applying rules (F 0.1) (F 1.1) produces, starting with Q,

(Is it true that)(SENT)

We now jump to a rule for sentencehood:

(F 10.1) SENT ⟶ (NI)(IS)(CONS)

(F 4.1) CONS ⟶ (POSPR2)(NI).

Substituting into (F 10.1) the result of rule (F 2.4) and into (F 4.1) the result of (F 2.6), we have SENT ⟶ (4) (IS) (POSPR2) (6). Applying (F 3.1) and (F 1.4) and combining, we have

"Is it true that 4 is above 6".

It is a well-formed sentence. With respect to Figure 2, it can be answered "Yes". With the help of rules (F 10.2), (F 1.2), (F 1.3) and (F 2.2), we could also have formed the more precise question:

"Is it true that in Figure 2 4 is above 6".

Had we used rule (F 10.3) rather than (F 10.2) in the above production process, we could have obtained:

"In Figure 2 is it true that 4 is above 6". This question is related to the one just above by a simple transformation. We can express it as:

(T 1.1) (QUAL)(QTRUE)(SENT)⟷(QTRUE)(QUAL)(SENT).

Given (F 10.4), (F 10.2), and if we have (T 1.1), we no longer need (F 10.3).

The set T of transformation rules are all like (T 1.1) and serve to render the questions generated by F more English-like and to avoid very awkward phraseology.

III. THE SOURCE LANGUAGE: FURTHER SPECIFICATIONS FOR ASSERTIVE QUESTIONS

By an assertive question we mean one generated by rule F 10.2. It begins with: "Is it true that...". Rule (F 4.1) has, so far, only enabled us to make assertions about individual objects. To be able to pose a question like, "Is it true that 4 is above every circle", we pick QUANT a (non-terminal) label for the class of phrases like "every circle". It is used in:

(F 4.2)	CONS ⟶ (POSPR2)(QUANT)	(QUANT is mnemonic for a clause with quantifiers)
(F 5.2)	QUANT ⟶ (QUE)(ANT)	
(F 5.3)	QUANT ⟶ (QUA)(ANT)	
(F 6.1)	ANT ⟶ VAR	(ANT is mnemonic for "antecedent")
(F 6.2)	ANT ⟶ (ANT)(CL)	(CL reads "Clause")
(F 7.1)	CL ⟶ (W)(IS)(CONS)	
(F 8.1)	VAR ⟶ object.	The word "object" is generic,

like the word "variable" for which VAR is mnemonic. We might have chosen to treat "object" as a one-place predicate. Since our universe of discourse however consists only of simple geometric figures which are individually labelled, the word object is generic to these labels. That is, 1, 2, 3, ... are specific to the word "object".

We now add to our previously started partial list of rules that relate to the terminal vocabulary- (F 1.X) for function words and (F 3.X) for predicates:

(F 1.5) QUE ⟶ an (the existential quantifier)

(F 1.6) QUA ⟶ each (the universal quantifier)

(F 1.7) NOT ⟶ not

(F 1.8) AND ⟶ and

(F 1.9) OR ⟶ or

(F 1.10) IF ⟶ if

(F 1.11) THEN ⟶ then

(F 1.12) W ⟶ which

This list will be completed in the section where the additional ones are used.

So far, we could, from the sequence of rules (F 10.1), (F 2.1), (F 1.4), (F 4.2), (F 3.1), (F 5.2), (F 1.5), (F 6.1), and (F 8.1) form the sentence (starting with SENT): "1 is above an object." If we wanted to improve this stylistically, we would introduce the transformation "an object" ⟶ "something". Next, we introduce more predicates:

(F 3.2) POSPR2 ⟶ below (another 2-place position predicate)

(F 3.3) POSPR2 ⟶ to the left of

(F 3.4) POSPR2 ⟶ to the right of

(F 3.5) POSPR2 ⟶ enclosing

(F 3.6) POSPR2 ⟶ inside

The reader can readily verify that from SENT, with the help of rules (F 10.1) and (F 4.2), and other rules we can form sentences like:

S_1: 1 is above each object

S_2: 1 is above each object which is enclosing 2.

S_3: 1 is below each object which is inside each object which is above 2.

S_4: 1 is below an object which is to the left of each object which is enclosing an object.

It is easy to see that an infinite number of well-formed sentences can thus be generated, because there is no restriction on how often rules (F 6.2), (F 7.1) and (F 4.2) can be reapplied. In a figure with a finite number of objects, only a finite number, however, can be materially true, if redundancies are not counted. The clause "object which is 1" can obviously be replaced by "1". Some sentences express impossible configurations like "1 is above each object," because 1 cannot be above itself.

(F 3.7)	DKPR2 ⟶	darker than	(a 2-place intensity predicate)
(F 3.8)	DKPR2 ⟶	lighter than	
(F 3.9)	SZPR2 ⟶	larger than	(a 2-place size predicate)
(F 3.10)	SZPR2 ⟶	smaller than	
(F 3.11)	SHPR1 ⟶	triangular	(a 1-place shape predicate)
(F 3.12)	SHPR1 ⟶	rectangular	
(F 3.13)	SHPR1 ⟶	circular	

Quite a variety of interesting sentences can be generated at this point, but they are stylistically awkward. The following transformations are of general value in producing more English-like sentences, and they improve the sentences producible so far.

T 1.2. (QUAL)(QTRUE)(QUANT)(IS)(CONS)⟷(QUAL)(IS)(QUANT)(CONS)

Example: "In Figure 1 is it true that each circle which is inside a triangle above a rectangle" ⟷ "In Figure 1 is each circle which is inside a triangle above a rectangle".

(T 2.1) object which is $\left\{\begin{matrix}\text{circ}\\ \text{rectang}\\ \text{triang}\end{matrix}\right\}$ular$\leftrightarrow\left\{\begin{matrix}\text{circ}\\ \text{rectang}\\ \text{triang}\end{matrix}\right\}$ le

(T 2.2) not an$\leftrightarrow$no

(T 2.3) not is it true that$\leftrightarrow$is it false that

(T 2.4) (QTRUE)(QUAL) an object is$\leftrightarrow$is there an object (QUAL)

(T 2.15) an $\left\{\begin{matrix}\text{circle}\\ \text{rectangle}\\ \text{circle}\end{matrix}\right\}\leftrightarrow$a $\left\{\begin{matrix}\text{circle}\\ \text{rectangle}\\ \text{triangle}\end{matrix}\right\}$

The following formation rules permit us to enrich our class of questions by incorporating the main devices of ordinary propositional logic.

(F 4.3) CONS $\longrightarrow$ (NOT)(CONS)

Together with (T 2.2), this rule could generate "above no object", "above no circle".

(F 4.4) CONS $\longrightarrow$ SHPR1

(F 4.5) CONS $\longrightarrow$ (CONS)(AND)(CONS)

(F 4.6) CONS $\longrightarrow$ (CONS)(OR)(CONS)

(F 9) IMP $\longrightarrow$ (ANT)(IS)(CONS); IMP is mnemonic for "implication", and if it is recalled that ANT suggests "antecedent" and CONS "consequent", this rule will be seen to stand out as one of the most essential and powerful.

We finally augment our rules for sentence formation:

(F 10.5) SENT $\longrightarrow$ (SENT)(AND)(SENT)

(F 10.6) SENT $\longrightarrow$ (SENT)(OR)(SENT)

(F 10.7) SENT $\longrightarrow$ (NOT)(SENT)

(F 10.8) SENT $\longrightarrow$ (IF)(SENT)(THEN)(SENT)

(F 10.9) SENT $\longrightarrow$ (QUE)(IMP)

(F 10.10) SENT $\longrightarrow$ (QUA)(IMP)

The following is a sample of 10 questions which can be generated from the rules stated up to this point.

E.1. In Fig. 1 is each circle which inside a triangle above a rectangle

E.2. Is there an object in Fig. 1 which is inside a circle which is inside a rectangle

E.3. Is it false that in Fig. 1 a circle is above each triangle

E.4. Is it true that in Fig. 1 no circle which is larger than a circle inside a triangle is to the left of a triangle which is above a rectangle

E.5. If each circle is larger than 2 then is it true that in Fig. 1 each circle is above a triangle.

E.6. Is it true that in Fig. 1 if each triangle is above a circle then if each triangle is inside a rectangle than a circle is inside a circle then a circle is larger than 3.

E.7. Is it true that in Fig. 1 a triangle is above a circle or a triangle is below a circle.

E.8. Is it true that in Fig. 1 each circle is inside a triangle and each circle is above each triangle.

E.9. Is it true that in Fig. 1 each rectangle which is inside a circle and to the left of a triangle which is inside a circle is larger than 3.

E.10. Is it true that in Fig. 1 each circle which is larger than 3 and smaller than 4 is inside a circle which is larger than 4 and smaller than 3.

Note that in E.9 we cannot specify whether the triangle is to be inside the same circle as the rectangle. To do this we should say "inside the circle" or "inside that circle" in place of the second occurrence of "inside a circle." This is an important device which has been dealt with by Bohnert (5).

Note that E.10 can be answered "No" as if it were an analytic sentence. To transform E.10 into an analytic sentence in which a logical contradiction is formally derivable, it is necessary to substitute formal definitions for "inside," "larger than" and "smaller than." One way to do this is to introduce special "inference rules" which are implicit in the data structure or in the algorithms for searching the data structures. For example, each of the 10 dyadic predicates introduced so far are transitive.

Rule 1: Let p denote any such predicate, e.g. "above." If x, y, z denote any three objects, and "x is py," "y is pz" are both true, then "x is pz" is also true.

Rule 2: If "x is inside y" and "x $\begin{bmatrix}\text{is larger than}\\ \text{is enclosing}\end{bmatrix}$ z" are both true, and x, y, z are all three circles or rectangles or triangles, then "y $\begin{bmatrix}\text{is larger than}\\ \text{is enclosing}\end{bmatrix}$ z" must also be true.

Rule 3: If "x is inside y" and "y is qz" where q stands for any of the ten predicates except "darker than," "lighter than," "larger than," "enclosing," then "x is qz" must be true.

If sentences derivable from these rules are combined by AND the (declarative) sentence part of an assertive question-sentence can be shown to be contradictory. That is, SENT has the form (s)(AND)(NOT)(s) where s has the form SENT.

IV. DOMAIN UPON WHICH QUESTION-ANSWERING PROGRAMS OPERATE

The algorithm being described produces, for each question in the source language, a flow diagram for a computer program. The computer program, if run, would search a data base for the correct answer to the question. The answer is obtained either by direct lookup or by inference from what can be looked up. The rules of inference are embedded in the rules which translate questions of the source language into flow diagrams.

To fix ideas, let us adopt a particular symbolic form for representing stored data as seen by a programmer. The data base consists of a list of descriptions of labelled figures. The figures are labelled - numbered - in the temporal order in which they are added to update the data base. Each figure can be described by several different methods. One is graphical, for easy visualization, as Figs. 1 and 2 at the beginning of this paper.

To describe the essential features of such a diagram completely enough to make possible the answer of all questions in the source language, we must adopt some representation. The choice of representation, particularly with regard to its effectiveness and efficiency for updating and answering of questions requiring inferences is a fundamental problem (3) (39) but not focal to this paper, in which we aim at probing the non-syntactic boundaries of our source language. The representation for storing diagram descriptions we chose takes advantage of the fact that all the two-place relations we used (is above, is to the right of, is inside, is darker than, is larger than) are transitive. To make this precise, let us symbolize, to indicate to a programmer how to encode it for storage, our predicates in the terminal vocabulary as follows:

H '(i,j) = "i is immediately to the left of j" = "j is immediately to the right of i"

H (i,j) = "i is somewhere to the left of j"

E H (i,j) = "i is not to the left of and not to the right of j"

V '(i,j) = "i is immediately above j" = "j is just below i."

V (i,j) = "i is somewhere above j"

E V (i,j) = "i is not above and not below j"

I '(i,j) = "j is immediately inside i" = "i is just enclosing j"

I (i,j) = "i is enclosing j" = "j is somewhere inside i"

E I (i,j) = "i is not enclosing and not inside j"

D '(i,j) = "i is just a shade darker than j"

D '(i,j) = "i is darker than j"

E D (i,j) = "i is not darker than and not lighter than j"

S '(i,j) = "i is just larger than j"

S (i,j) = "i is larger than j"

E S (i,j) = "i is not larger and not smaller than j"

R (i) = "i is rectangular"

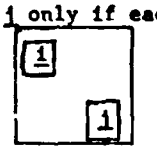
T (i) = "i is triangular"

C (i) = "i is circular"

To completely represent Fig. 2, we would use the expression:

FIG. 2 = I'(2, H'(3, V'(I'(4,5),6))) & EH (4,6) & T(3) & R(4) & C(5) & T(6)

The & means "and"; it takes all of these statements to describe the figure to within correct horizontal and vertical alignments. That is,  and  are considered identical, and can both be represented

by H'(i,j) & EV (i,j). We would call i above j only if each point of i is above each point of j. We would represent 

by H'(i,j) & V'(i,j).

To answer the question, "In Fig. 2, is it true that 5 is inside 4", we interpret "inside" to be either "just inside" or "somewhere inside." We search the data base for a description of Figure 2, by going through 2 as an index term which points to the location where the description is stored. The description, a coded version of Fig. 2, is retrieved and scanned for I'(4,5). Since this is readily found, the search ends with a positive answer.

To answer the question, "In Fig. 2, is it true that 5 is above 6" requires a more subtle search through the expression in Fig. 2. The search is completed with a positive answer either if we find: (a) V'(5,6) or (b) V(5,6); rules of inference applying to these representations also permit us to add conditions like: (C) V'(I'(i,5),6) as well as many other conditions which follow from an inference rule like: "any object inside an object above a third object is above that third object." All these conditions are equivalent to V(5,6).

We assume that subroutines for searching a string like Fig. 1 for the truth or falsity of V(i,j) are stored, can accept as input any such string, and produce as output either:

"Yes, V(i,j) is true for input string FIG. 1" or

"No, V(i,j) is not true for input string FIG 1."

We represent this subroutine by the following fragment of a flow diagram:

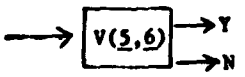


Fig. 1.

The domain upon which the question-answering program operates, then, is the set of all acceptable strings like Fig. 1. The following simple rules of formation can characterize this set:

S ⟶ E/P1(N)/S&S, when S is mnemonic for the set of well-formed strings. The slash, /, means alternatives, i.e., "or."

E ⟶ P2(A,A)

P2 ⟶ H'/V'/I'/EH/EV/EI as defined before

A ⟶ N/E

N ⟶ 1/2/3/4/5/...

P1 ⟶ R/T/C as defined before.

We could develop a grammar for generating English-like declarative sentences corresponding to these strings, and these declarative sentences would all be much simpler than the questions in our source language because they need only provide minimal irredundant descriptions of the figures.

The language generated by this simple grammar is infinite because both E and S can appear infinitely often through the iterated use of rules (1c), (2) and (4b). That is, configurations of an arbitrarily large number triangles, circles and rectangles of varying size and intensity combined in an arbitrarily large and complex number of arrangements can constitute any one figure being stored. The data base is a stored corpus of such figures.

V. TRANSLATING ASSERTIVE QUESTIONS INTO FLOW DIAGRAMS

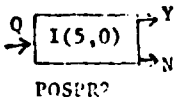
Given a question in the source language, the first step of the algorithm is to parse the question. This means the construction of a phrase-marker or labelled bracketting of the question indicating which rule of G is applied when. Let us illustrate with the question:

Q = "Is it true that in Figure 2 4 is enclosing 5."

We scan Q from left to right until we find a string which matches an argument in Table I, looking for the longest possible match (11), (22). Thus, though we can match "is" as the 4th entry in Table I, we match on "is it true that", the 2nd item in Table I. Table I tells us to apply rule (F 1.1), which is QTRUE ⟶ is it true that. We thus bracket "is it true that" in Q and indicate that it may have been generated from QTRUE.

We proceed with our left-to-right scan, to bracket "in" and indicate that it was produced from IN by rule (F 1.2). We indicate that this was our second step by the circled 2. As we proceed to step 7 in this way we encounter rule (F 3.5), POSPR2 ⟶ enclosing, we note* in Table I that there is an accompanying translation rule TR 3.5. It is to form:

(TR 3.5)

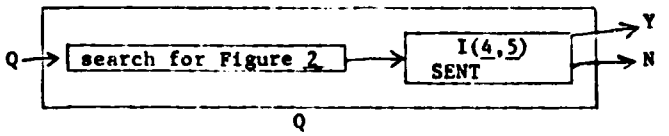


POSPR2 is the name of this box. Its input is labelled Q and represents a control signal to initiate the operation of searching a specified string for the truth of V(s,o), after s and o have been specified. The output is a decision, a conditional transfer. It is Y or N, depending on whether

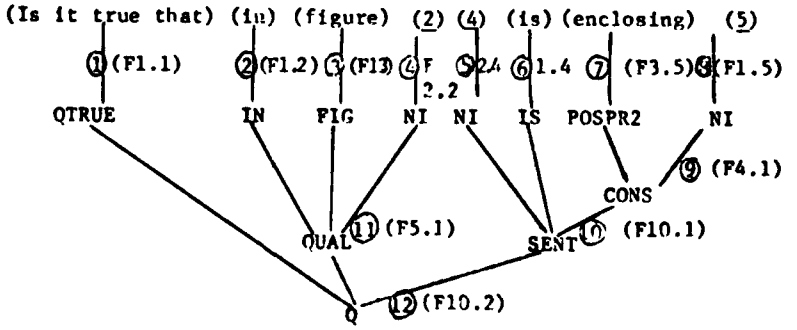
*Table I lists the translation rule for only "enclosing," as an example, but does not indicate which translation rules goes with each word or phrase, if any.

V(s,o) is true or false. Control is transferred from POSPR2 to the box to which the Y or N line, whichever is activated, points.

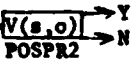
After we scanned Q from left to right and obtained a string Q, of non-terminal symbols from which Q may have been generated, we scan Q' from right to left. In step 9, the box previously named POSPR2 is renamed CONS and o is set to 5. At step 10, the box is again renamed SENT and we set s to 4. At step 11, we form the box → search the data base for Figure 2 → . At step 12, these two boxes are joined to produce:



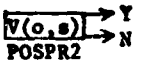
The entire flow diagram is now, in step 12, by (F 10.2), is labelled Q.



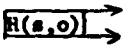
Next we describe the translation rules accompanying the appropriate formation rules.

TR 3.1 (with rule (F 3.1)): Form Q→ 

("Is s above o")

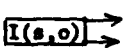
TR 3.2 (with rule (F 3.2)): Form ⟶ 

("Is s below o")

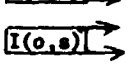
TR 3.3 Form ⟶ 

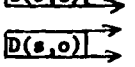
("Is s to the left of o")

TR 3.4 Form ⟶ 

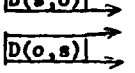
TR 3.5 Form ⟶ 

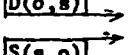
("Is s enclosing o")

TR 3.6 Form ⟶ 

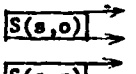
TR 3.7 Form ⟶ 

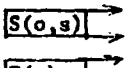
("Is s darker than o")

TR 3.8 Form ⟶ 

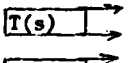
TR 3.9 Form ⟶ 

("Is s larger than o")

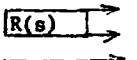
TR 3.10 Form ⟶ 

TR 3.11 Form ⟶ 

("Is s triangular")

TR 3.12 Form ⟶ 

("Is s rectangular")

TR 3.13 Form ⟶ 

(Is s circular")

There are no translation rules with (F 1.X) except for "an" and "each," For (F 2.i), which is NI↔i, we have rules TR 2.i, for which we think of i as a value of NI. We number the values of NI as these occur in Q',from left to right. TR 2.i tells us to store the value of NI in a push-down list, so that the right-most occurrence of NI is on top. This push-down is used in many other translation rules.

TR 4.1 (goes with CONS ⟶ (POSPR2)(NI)). POSPR2 is the name of a box formed by one of the rules TR 3.1 - TR 3.10.

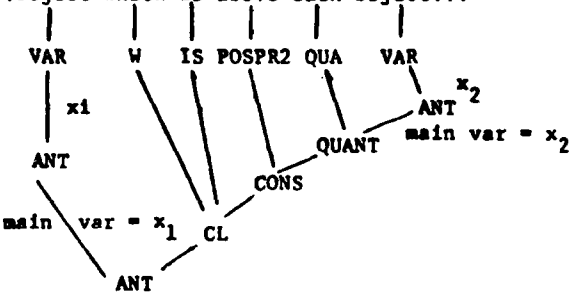
We now substitute for o the value of NI and we rename the box CONS.

To introduce the other translation rules, it is best to proceed in a certain order, beginning with:

TR 8.1 (goes with VAR⟶object). Here, as in TR 2.1, we form a push-down list x_1, x_2, x_3,..., assigning x_1 for the left-most occurrence of the word "object" as we scan Q from left to right; x_2 for the second occurrence of "object" in the left-to-right scan, etc.

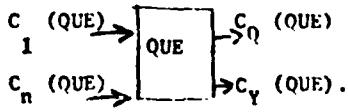
TR 6.1 (goes with ANT⟶VAR). When rule F 6.1 is applied to indicate that ANT generated VAR, we assign to ANT the variable of the top of the push-down list. We call this its main variable. To illustrate, consider the phrase (ANT), "...object which is above each object..."

The concept* of "main variable" will be used later in binding all the variables by quantifiers.



TR 6.2 (goes with ANT⟶(ANT)(CL)). If the box marked CL has s in it, substitute for s the main variable of ANT, say x_k. Relabel the new box ANT. Its main variable is also x_k.

TR 7.1 (goes with CL⟶(W)(IS)(CONS)). Substitute CL for CONS as the label of the box and make the main variable of CL the same as that of CONS. First, we return to

TR 1.5 (with QUE ⟶ an). Form the box 

This is an abbreviated version of those parts of the flow diagram due to the presence of a quantifier in a search operation. In more detail this box is:

*This concept, like several others pertaining to translation, from questions to the predicate calculus, is described in unpublished work by S. Amarel.

Fig. IIA

The input C_1(QUE) is a control signal which starts the QUE routine by initializing a search through a previously specified list. The input C_n(QUE) is the control signal to initiate a test to determine if search should continue or cease.

The output C_Q(QUE) is the signal which transfers the input data - the query specification - to a search routine, usually to a box labelled ANT. The output C_Y(QUE) is a signal indicating that as the search routine to which QUE is coupled succeeded in "matching" the input data of the question with something in the list being searched.

TR 1.6 (with QUA→each). same as TR 1.5.

TR 5.2 (with QUANT⟶(QUE)(ANT)). The box labelled ANT, like all boxes associated with a search routine have one input, Q(ANT) and two outputs Y(ANT) and N(ANT); Q signals the start of searching; Y signals success and N failure in searching. The coupling between QUE and ANT to form QUANT is specified by:

Q(QUANT) = C_1(QUE Q(ANT) = C_Q(QUE)

Y(QUANT) = Y(ANT N(ANT) = C_n(QUE)

N(QUANT) = C_Y(QUE) Main variable of QUANT = Main var. of ANT.

This may be clarified by the diagram.

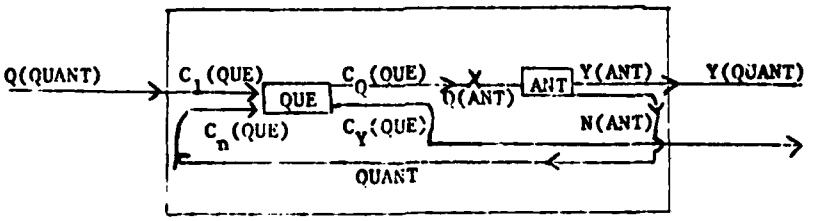


Fig. IIB

In terms of the blow-up for QUE, this is

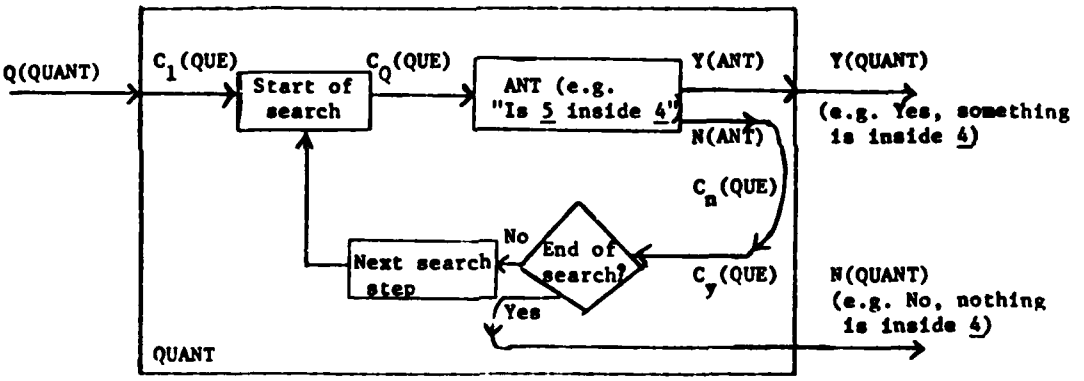


Fig. IIC

TR 5.3 (with QUANT ⟶ (QUA)(ANT))

Q(QUANT) = C_1(QUA) Q(ANT) = C_Q(QUA)

Y(QUANT) = C (QUA) Y(ANT) = C_n(QUA)

N(QUANT) = N(ANT) Mean Variable of QUANT = Main Var. of ANT

We show the connections in order to explain the next rule, TR 4.2, as well.

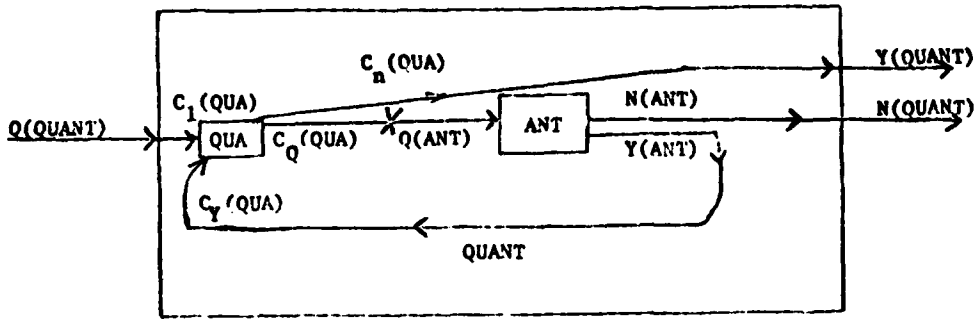


Fig. IID

TR 4.2 (with CONS $\longrightarrow$ (POSPR2)(QUANT)). Here the connection marked X in the above diagram of the QUANT box which was just made in combining QUA with ANT, is broken to allow an insertion. This also applies to the QUANT diagram for (QUE)(ANT). We replace the coupling expressed by Q(ANT) = C_Q(QUA) (or = C_Q(QUE)) by Q(POSPR2) = Y(POSPR2). In addition we have:

Q(CONS) = Q(QUANT) Y(CONS) = Y(QUANT)

N(CONS) = N(QUANT) = N(POSPR2)

Replace o in POSPR2 by the main variable of QUANT, and make this also the main variable of CONS.

TR 4.3 (with CONS $\longrightarrow$ (NOT)(CONS))

This is a simple relabeling or inversion of output leads.

Y((NOT)(CONS)) = N(CONS); N((NOT)(CONS)) = Y(CONS)

Q((NOT)(CONS)) = Q(CONS)

An analogous rule, T 10.7, also holds for SENT $\longrightarrow$ (NOT)(SENT)

TR 4.4 (with CONS $\longrightarrow$ SHPR1)

Change the label of the box from SHPR1 to CONS and let the main variable of CONS be that of SHPR1; namely s.

TR 4.5 (with CONS ⟶ (CONS)(AND)(CONS)).

This rule is identical with TR 10.5, that corresponding to SENT ⟶ (SENT)(AND)(SENT) and is obvious to anyone familiar with propositional logic. It is best stated in a diagram, in which CS1 stands for either the first CONS or SENT on the right-hand-side of the formation rule, and CS2 stands for the second occurrence. CS is the result of combining (CS1)(AND)(CS2).

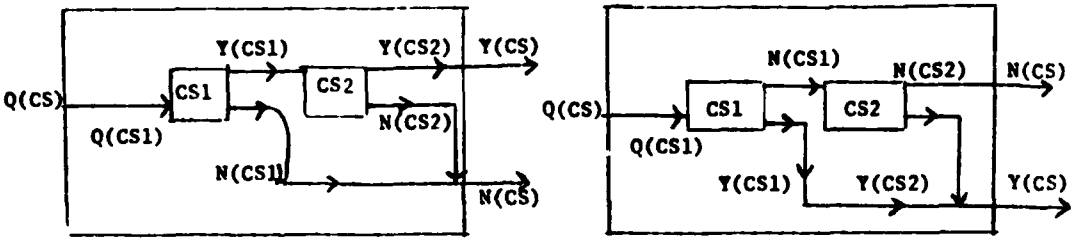


Fig. IIE

TR 4.6 (with CONS⟶(CONS)(OR)(CONS)). See diagram, identical with rule TR 10.6.

The next rule is most important.

TR 9 (with IMP⟶(ANT)(IS)(CONS)). The rule is:

Q(IMP)	= Q(ANT)	Y(ANT)	= Q(CONS)
Y(IMP)	= Y(CONS)	N(ANT)	= Y(CONS)
N(IMP)	= N(CONS)	Main Variable of IMP = Main. Var. of ANT	

and the main variable of IMP is s wherever s appears in CONS.

This corresponds to the usual definition of implication, as if IMP ⟶ (NOT)(ANT)(OR)(CONS).

We are now in a position to complete the flow diagram by giving the rules that go with forming sentences and assertive questions.

TR 10.9 (with SENT⟶(QUE)(IMP))

Q(SENT)	= C_1(QUE)	Q(IMP)	= C_Q(QUE)
Y(SENT)	= Y(IMP)	N(IMP)	= C_n(QUE)
N(SENT)	= C_Y(QUE)		

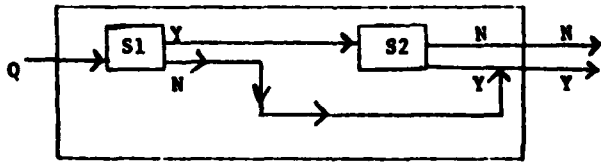
This rule should bind the only remaining variable.

TR 10.10 (with SENT⟶(QUA)(IMP)). Same as TR 10.9 with QUA in place of QUE

We need no translation rules for F 10.1 except to relabel the box from CONS to SENT; for F 10.4 the QTRUE prefix indicates merely that we have an assertive question, and that the algorithm developed up to this point applies. The translation rule to go with F 5.1 narrows the part of the data base to be searched by search routines. This rule then allows us to complete the flow diagrams when we apply F 10.2 or F 10.3.

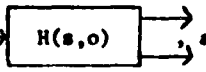
TR 10.8 (with SENT⟶(IF)(SENT)(THEN)(SENT))

This is best shown in the diagram, with S1 and S2 denoting the first and second occurrence of SENT to the right of the arrow in F 10.8. It is "not S1 or S2".



This concludes the set of translation rules to be used in forming a flow diagram as a by-product of parsing an assertive question. To summarize the algorithm: (1) the question is parsed; (2) with each

formation rule, starting first from left to right, then back from right to left, back and forth, a corresponding translation rule is applied; (3) the result of applying a translation rule inserts a box into the flow diagram, assigns variables and constants to the appropriate search routines; (4) when parsing is completed, so is the flow diagram; all variables are quantified by iterative scans.

All inferences are inherent in the representation of stored data, not in the source language of questions. Computing these inferences is implicit in search boxes, like 

and would be the subject of separate study.

Storing the translation rules so that they can be used to implement this algorithm somewhat resembles storing the rules of formation and transformation. The latter involve replacement instructions, such as "STRING 'PATTERN' = 'SUBSTITUTE'" in SNOBOL. The translation rules involve primarily the coupling of connections: The instructions would resemble those for wiring a circuit or a configuration of logic modules.

To illustrate how the algorithm works consider a typical assertive question. The diagrams show how the flow diagram builds up as we parse, Fig. III shows the phrase marker and Fig. IV shows the completed flow diagram for the program to answer the question.

VI. QUESTIONS INVOLVING "WHAT," "WHERE," AND "HOW"

With the preceding groundwork, we are now in a position to pursue our primary aim of showing that we can build upon our primitive algorithm so that it can handle questions in an enriched source language. The universe of discourse remains the same and the syntax is basically unchanged. The vocabulary is extended only slightly, but profoundly. In this section, we "enrich" the source language by admitting questions like, "In Figure 2 what object is darker than 3".

We first augment our basic vocabulary rules.

(F 1.13) WHA ⟶ what.

(F 1.14) WHE ⟶ where.

(F 1.15) HOW ⟶ how

(F 1.16) REL ⟶ related to

and our query-formation rules

(F 10.11) Q ⟶ (QUAL)(WHA)(IMP)

(F 10.12) Q ⟶ (WHA)(IS)(NI)

(F 10.13) Q ⟶ (WHE)(IS)(NI)

(F 10.14) Q ⟶ (HOW)(IS)(NI)(REL)(NI)

The translation rules accompanying these are:

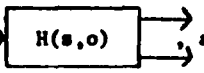
(TR 1.13) Form a box for WHA as for QUE

(TR 10.11) Treat (QUAL)(WHA)(IMP) as if it were (QTURE)(QUAL)(QUE)(IMP) except:

(a) replace the "No" answer by "No such object"

(b) replace the "Yes" answer by the label of the first object in the list being searched which satisfies the query specification.

formation rule, starting first from left to right, then back from right to left, back and forth, a corresponding translation rule is applied; (3) the result of applying a translation rule inserts a box into the flow diagram, assigns variables and constants to the appropriate search routines; (4) when parsing is completed, so is the flow diagram; all variables are quantified by iterative scans.

All inferences are inherent in the representation of stored data, not in the source language of questions. Computing these inferences is implicit in search boxes, like 

and would be the subject of separate study.

Storing the translation rules so that they can be used to implement this algorithm somewhat resembles storing the rules of formation and transformation. The latter involve replacement instructions, such as "STRING 'PATTERN' = 'SUBSTITUTE'" in SNOBOL. The translation rules involve primarily the coupling of connections: The instructions would resemble those for wiring a circuit or a configuration of logic modules.

To illustrate how the algorithm works consider a typical assertive question. The diagrams show how the flow diagram builds up as we parse, Fig. III shows the phrase marker and Fig. IV shows the completed flow diagram for the program to answer the question.

VI. QUESTIONS INVOLVING "WHAT," "WHERE," AND "HOW"

With the preceding groundwork, we are now in a position to pursue our primary aim of showing that we can build upon our primitive algorithm so that it can handle questions in an enriched source language. The universe of discourse remains the same and the syntax is basically unchanged. The vocabulary is extended only slightly, but profoundly. In this section, we "enrich" the source language by admitting questions like, "In Figure 2 what object is darker than 3".

We first augment our basic vocabulary rules.

(F 1.13) WHA ⟶ what.

(F 1.14) WHE ⟶ where.

(F 1.15) HOW ⟶ how

(F 1.16) REL ⟶ related to

and our query-formation rules

(F 10.11) Q ⟶ (QUAL)(WHA)(IMP)

(F 10.12) Q ⟶ (WHA)(IS)(NI)

(F 10.13) Q ⟶ (WHE)(IS)(NI)

(F 10.14) Q ⟶ (HOW)(IS)(NI)(REL)(NI)

The translation rules accompanying these are:

(TR 1.13) Form a box for WHA as for QUE

(TR 10.11) Treat (QUAL)(WHA)(IMP) as if it were

(QTURE)(QUAL)(QUE)(IMP) except:

(a) replace the "No" answer by "No such object"

(b) replace the "Yes" answer by the label of the first object in the list being searched which satisfies the query specification.

Example:

Each object which is above an object which is enclosing an object is to the right of an object which is enclosing an object.

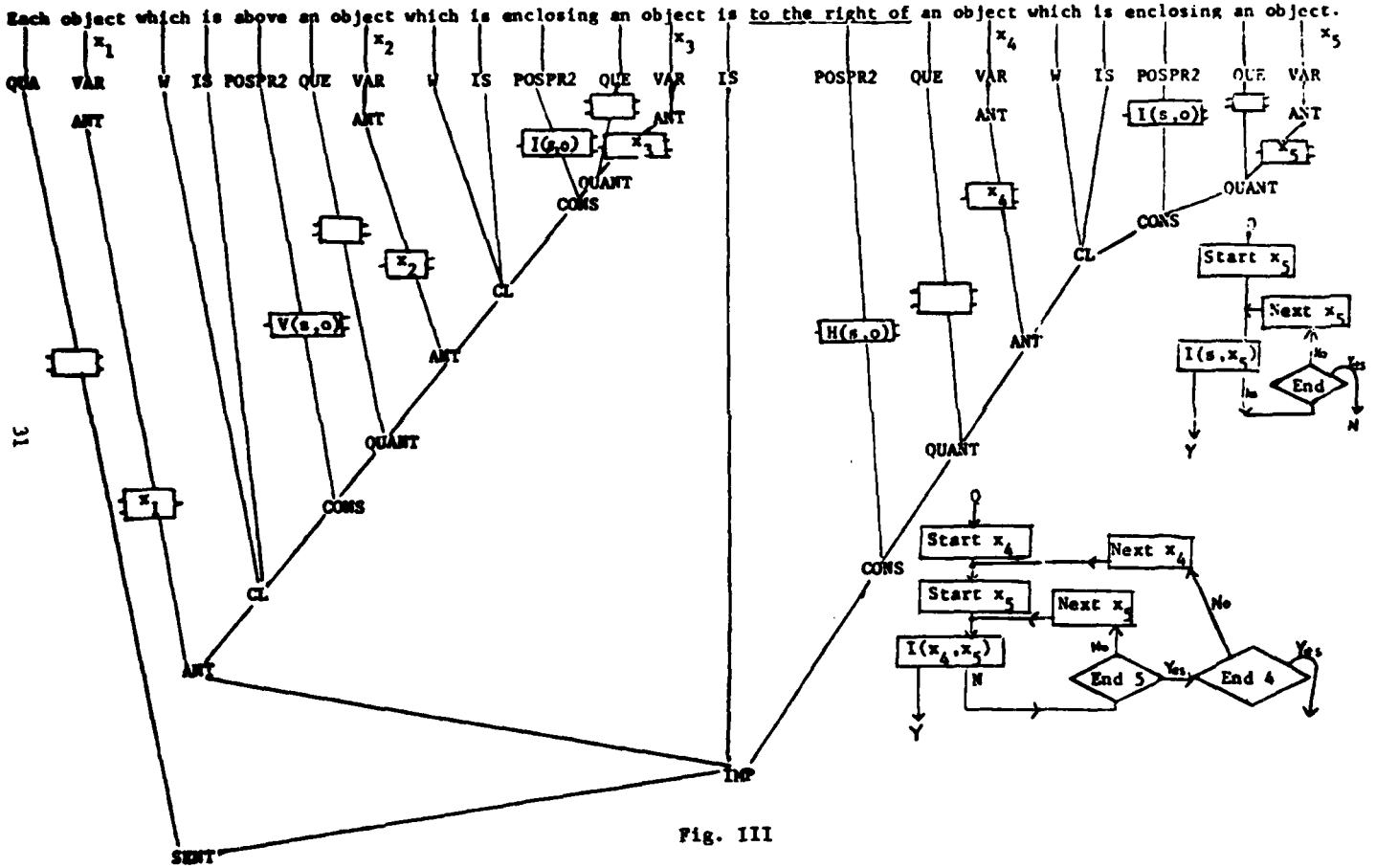


Fig. III

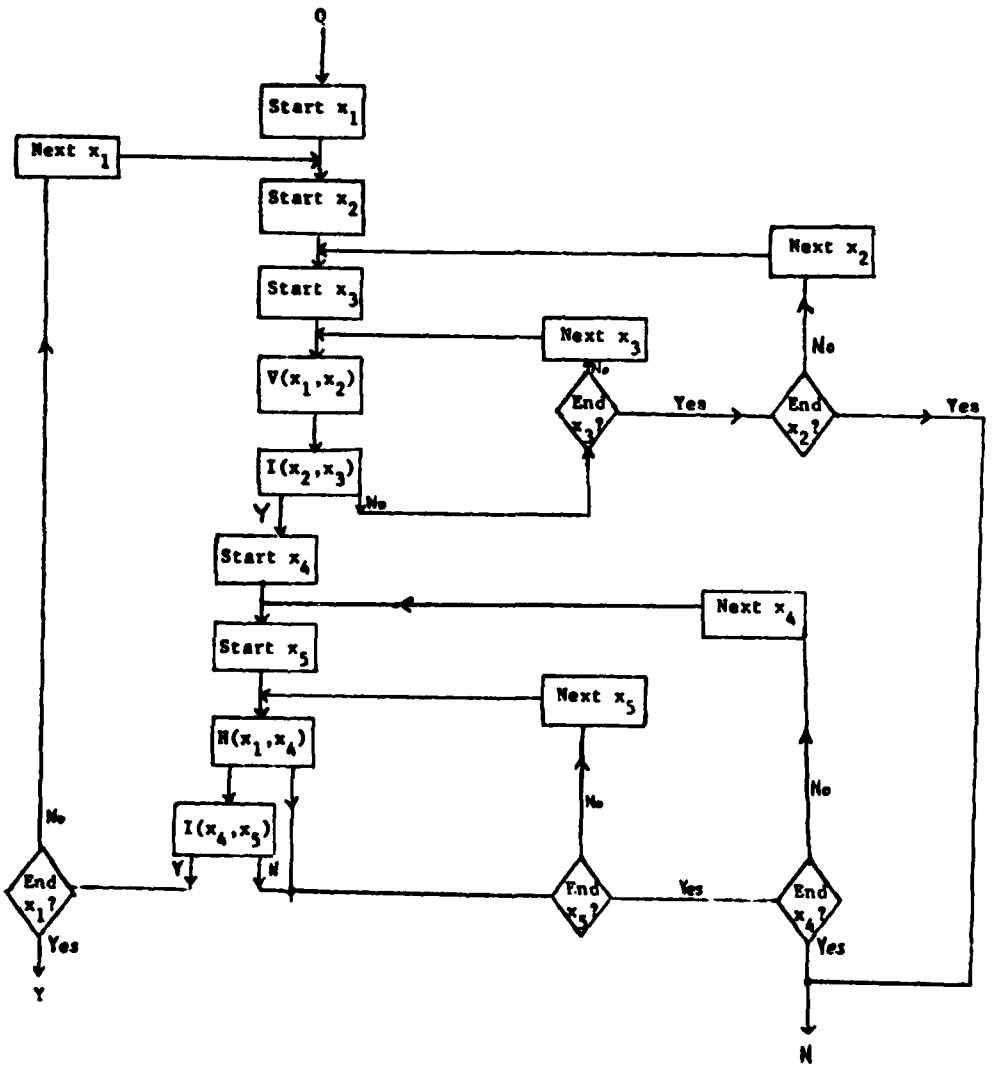

0
Start x_1
Next x_1
Start x_2
Next x_2
Start x_3
Next x_3
$V(x_1,x_2)$
End x_3?
Yes
End x_2?
No
Yes
$I(x_2,x_3)$
No
Y
Start x_4
Next x_4
Start x_5
Next x_5
$H(x_1,x_4)$
$I(x_4,x_5)$
No
No
Y
N
End x_1?
End x_5?
Yes
End x_4?
Yes
Yes
Y
N

Fig. IV

(c) we would also provide the answer as a complete declarative sentence, (QUAL)(IMP), in which the word "object" is replaced by the label of the search result.

Example: "In Fig. 2 what object is darker than 3" →

"In Figure 2, 5 is darker than 3". Certainly this declarative sentence is relevant to the question, whether or not it is correct.

English usage would give preference to "which object" over "what object," to connote that one of a list of objects is to be found, while "what" is more appropriate to, say, "What is 2." Only the attributes, size, shape and intensity, but not position, are considered relevant to this sense of "what." To handle such a query, we introduce the translation rule TR 10.12. It instructs to search the entire data structure, through an index, for the value of NI; then to read out all the sentences involving S, D, C, T or R (also using an index).

A sentence like "In Fig. 1 what is an object which if inside a circle is above a triangle" is first transformed to: "In Fig. 1 what object which is inside a circle is above a triangle." If this second question produces an NI, the value of this NI is used to form "what is NI." To effect this complex transformation we use:

T 1.3 (QUAL)(WHA)(VAR)(W)(IS)(CONS) → (QUAL)(WHA)(IS)(QUE)(VAR)(W)(IF)(CONS)

The question form which is relevant to positional attributes (POSPR2) involves "where."

TR 10.13. This translation rule is very similar to that for "what." As in the case of "what," it is possible to transform a question like "In Fig. 1 what object is inside a circle and above a triangle" into "In Fig. 1 what object is inside a circle and above a triangle" by rule T 1.3. As before,

the name (say i) of an object answering the above description is then used to form "Where is i," the answer to which is provided by statements like: H'(i,4), H'(7,i), V'(i,6), I'(i,5), i is in Fig. 1. The answer to "what is i" might look like: D'(i,11), D'(13,i), S'(i,3), C(i).

To handle like "In Fig. 1 [where / what] is each object which if inside a a circle is above a triangle," consider a transformation into "In Fig. 1 what objects which are inside a circle are above a triangle." This transformation, involving inflections, is beyond the scope of this article. The point, however, is to use QUA instead of QUE when processing this query and produce all objects satisfying the query specification, not just an arbitrary one - i.e., the first one encountered during list-searching.

If there is no object meeting the query specifications, the response is always "no such object." If the query had "the" in place of "an" or "each (e.g., "In Fig. 1 [where / what] is the object which...") there is an implied assertion that there exists a unique object meeting the query specification. If, in fact, there is more than one, then the query was underspecified. If none, the query was either over-specified or wrongly specified.

To deal with queries involving "when" and "how," change-sentences are introduced in the next section. The introduction of "how many" and "how much" leads to a development of sufficient scope to warrant its being reported as a separate paper. But there is one sense of "how" which does not require the use of change-sentences, exemplified by: "How is 2 related to 3." To this we turn next.

TR 10.14. Search an index for the values of the two NI, say i,j. Then read out all stored "sentences" involving both, e.g.: H'(i,j), EV(i,j), C(i), T(j), D(i,j). Questions in which NI is replaced by QUANT are ambiguous.

A question like, "How is 2 related to each triangle which is inside a circle," can be dealt with in two ways. We first form the question, "What triangles are inside circles," to get possible answers i_1, i_2,.... We now interpret the original question as either "How is 2 related to i_1 and i_2 and ..." or as "How is 2 related to i_1 and how is 2 related to i_2 and ..." The latter we can handle with the algorithm developed so far. The former requires special routines such as developed in the ANIP system (26).

VII. EXTENDING THE SOURCE LANGUAGE TO QUESTIONS ABOUT CHANGE

Consider the collection of figures shown below:

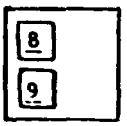


Fig. 1

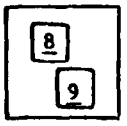


Fig. 2

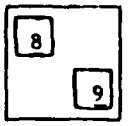


Fig. 3

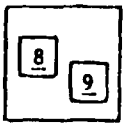


Fig. 4

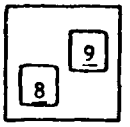


Fig. 5

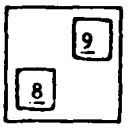


Fig. 6

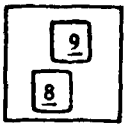


Fig. 7

Because the same set of individuals appear in different relations in all these figures, we can speak of changes in these relations. The number of each figure corresponds to the time that the description of a particular configuration is recorded for storage. Thus, at time 1, the data V'(8,9) & H'(8,9) & R(8) & R(9) is entered at address 26; at time 2, the same data is entered at address 27; at time 3, the data, V'(8,9) & H'(8,9) & R(8) & R(9) is entered at address 28, etc. Implicit in this data structure and/or the procedure for searching it are the statements: "Fig. i was recorded immediately before Fig. i + 1" and "Fig. i was recorded (sometime) before Fig. j" if $j>i$. The predicate "before" is transitive.

Before introducing tense into the syntax of the English-like queries, it is useful to introduce the phrases "changes occur," "from where" and "to where" as well as the notion of a "standard sentence." We first formally introduce these as terminal vocabulary elements:

F 1.17 CHOC → changes occur

F 1.18 FRWH → from where

F 1.19 TOWH → to where

Next we introduce a new type of sentence:

F 10.15 SENT → (CHOC)(PRE)(POST)

F 11.1 PRE → (FRWH)(SENT)

F 11.2 POST → (TOWH)(SENT)

F 11.3 PRE → (FRWH)(QUAL)(SENT)

F 11.4 POST → (TOWH)(QUAL)(SENT)

The translation rule corresponding to F 11.3 and F 11.4 is to search a list in the location specified in QUAL as if we had (QTRUE)(QUAL)(SENT). The translation rule with F 10.15 is to combine PRE with POST by conjunction, as if it were (PRE)(AND)(POST).

We next introduce the interrogative form with the help of F 10.4. This produces a sentence like: "Is it true that changes occur from where in Fig. 1 8 is above 9 to where in Fig. 7 9 is above 8." The answer to this question is "Yes" if and only if the answers to both "Is it true that in Fig. 1 8 is above 9" and "Is it true that in Fig. 7 9 is above 8" are "Yes."

With rules F 11.1 and F 11.2, the translation rules require a search of the entire corpus of stored data for a figure in which the data specified by SENT is stored. Thus, to answer "Is it true that changes occur from where 8 is above 9 to where 9 is above 8" it is necessary to locate Figs. 1, 2, or 3 for PRE and 6 or 7 for POST. Any of these six pairs would produce a "Yes" answer.

To compare two sentences describing changes we introduce:

F 1.20 WHN -- when

F 1.21 AFT -- after

F 1.22 BEF -- before

F 1.23 WHI -- while

These new elements are used as follows:

F 10.16 SENT -- (SENT)(WHN)(SENT)

This will apply only when SENT on the right-hand side was formed by rule F 10.15. This involves checking that the same pair of figure numbers occur in both sentences. For example, consider the query: "Is it true that changes occur from where in Fig. i 30 is above 40 to where in Fig. j 40 is above 30 ____x____ changes occur from where in Fig. u 35 is inside 45 to where in Fig. v figure 35 is to the left of 45."

The answer is "Yes" if:

x = "when"; i = u, j = v;

x = "after"; i > v

x = "before"; j < u

x = "while"; u ≤ i, v ≥ j

In all cases, i < j and u < v.

Each of the time-relations couples the two sentences on the right-hand side of a rule like F 10.16 according the above 4 translation rules. The other three corresponding formation rules are similar in form to F 10.16 with AFT, BEF and WHI in place of WHN.

To deal with a query like, "When do changes occur from where 30 is above 40 to where 40 is above 30," we introduce:

F 1.24 DO → do

F 10.17 Q ⟶ (WHN)(DO)(SENT)

With this rule goes a search for two figures containing the data specified in the PRE and POST part of SENT on the right-hand side. If the first of the two figures is Fig. i and second Fig. j, the answer would be "Changes

from where 30 is above 40 to where 40 is above 30 begin at i and end at j." We will not introduce the necessary complexities to produce such a complete sentence as the answer at this point, but accept (i,j) as an answer.

To introduce verbs and tense inflections consider:

F 1.25 STDSENT → this sentence is uttered

F 10.18 SENT ⟶ (SENT)(WHN)(STDSENT)

Incoming data, which is to update the total data structure, is grouped in expressions describing labeled figures. The labels reflect when these figure descriptions were stored. A question-sentence posed to the data system can refer to two such figure labels, one corresponding to the last figure stored prior to the question being posed, the other to the first figure stored after the query is posed. In the following query-sentence, the pronoun "this" refers to the total (largest) sentence of which that phrase in a clause: "Is it true that changes occur from where in Fig. 30 8 is above 9 to where in Fig. 31 8 is inside 9 when this sentence is uttered." With this entire sentence go two figure numbers, i and j. The query is answered "Yes" if i = 30 and j = 31; otherwise "No." The query sentence is not regarded as part of the figures to which it corresponds. We now extend this scheme:

F 10.19 SENT⟶(SENT)(AFT)(STDSENT)

F 10.20 SENT⟶(SENT)(BEF)(STDSENT)

F 10.21 SENT⟶(SENT)(WHI)(STDSENT)

To make the resulting queries more English-like, we add several transformations:

T 2.5 Changes occur from where x is smaller than y to where x is larger than y $\longrightarrow$ x increases from where x is smaller than y to where x is larger than y.

T 2.6 Changes occur from where x is larger than y to where x is smaller y $\longrightarrow$ x decreases from where x is larger than y to where x is smaller than y.

T 2.7 Changes occur from where x is above y to where x is below y $\longrightarrow$ x moves from where x is above y to where x is below y.

T 2.8 (x)(Vs)(PRE)(POST)(WHE)(STDSENT) $\longrightarrow$ (x)(Vs)(PRE)(POST)

Here V stands for "increase," "decrease," "move." While (x)(y) has stood for the concatenation of x and y with a space in between, Vs stands for concatenating s right after V without a space.

T 2.9 (x)(Vs)(PRE)(POST)(AFT)(STDSENT) $\longrightarrow$ (x)(will)(V)(PRE)(POST)

T 2.10 (x)(Vs)(PRE)(POST)(BEF)(STDSENT) $\longrightarrow$ (x)(Vd)(PRE)(POST)

T 2.11 (x)(Vs)(PRE)(POST)(WHI)(STDSENT) $\longrightarrow$ (x)(is)(V$\not$ing)(PRE)(POST)

The $\not$ indicates that if V ends in e, this e is to be deleted prior to concatenating "ing" without a space the V stripped of e. We treat "will", "d,", "$\not$ing" as terminal vocabulary elements but do not produce them from non-terminal symbols.

To deal with a question like, "How did 2 move from where in Fig. 1 2 is above 3 to where in Fig. 4 2 is below 5," we first consider a transformation from the variant of the sentences produced by T 2.10

T 2.12 (HOW(x)(Vd)(PRE)(POST) $\longrightarrow$ (HOW)(did)(x)(V)(PRE)(POST)

The rule for flow-diagram construction is to treat the question as if it were a search for all figures between Fig. 1 and Fig. 4 if these are specified as part of PRE and POST, and a search for all figures

from Fig. 1 to Fig. 2 (inclusive) if these are not specified in the question. The answer is to be in the form of a step-by-step description of how the motion occurred, insofar as data about all these steps is recorded.

Similarly, we have:

T 2.13 (HOW)(x)(is)(Ving)(PRE)(POST) ⟶ (HOW)(is)(x)(Ving)(PRE)(POST)

T 2.14 (HOW)(x)(will)(V)(PRE)(POST) ⟶ (HOW)(will)(x)(V)(PRE)(POST)

It is not necessarily the case that all consecutive figure numbers between the one (explicitly or implicitly) of PRE and that of POST pertain to the question. All of them must, however, be examined. If, in two successive figures, all the three objects mentioned in the above question-example have not changed their relative position, then this need not be reported in the answer. The answer need only report what changes took place and when.

VIII. CONCLUSION

This paper describes a procedure for translating English-like questions about simple pictures into flow diagrams for computer programs which would answer the question. What is new and important about this algorithm is that it can be cumulatively augmented to extend the English-like source language of questions, not in syntax or domain of discourse, but in vocabulary representing fundamental concepts common to all questions.

It would be ideal to be able to state, as a theorem, the precise limits on the source language. In the conventional sense, the grammar G presented here specifies the source language L_G syntactically, but this is not very pertinent. In the absence of techniques for proving a theorem about the extent to which L_G includes significant question-types, a sampling of sentences within (and outside) the content scope of the algorithm will have to serve as an adequate means of conveying to the reader an impression of the extent of this scope.

The following set of ten sentences, combined with sentences E 1, ..., E 10 in the text, provide a sampling of positive instances.

E 11 How did 20 move from where in Fig. 1 20 is above 30 to where in Fig. 5 20 is below 30.

E 12 How is 5 moving from where 5 is inside 6 to where 5 is enclosing 6.

E 13 How will 6 move from where in Fig. 1 6 is to the left of 7 to where 6 is to the right of 7.

E 14 How is a triangle which is inside 5 increasing from where it is just smaller than 7 to where it is just larger than 8.

E 15 When did changes occur from where 1 is just darker than 2 to where 1 is just darker than 3.

E 16 Is it true that a triangle which is larger than each triangle which is inside a circle increases from where it is smaller than a triangle inside a rectangle to where it is greater than a triangle which is inside a triangle.

E 17 Where is a circle which is inside a circle which is above a rectangle which is inside a triangle which is larger than each triangle which is above each circle.

E 18 How is a triangle inside a circle related to a triangle which is not inside an object.

E 19 Is it true that if Fig. 1 no object is inside an object.

E 20 Is it true that in Fig. 1 if no object is inside an object then in Fig. 1 no object is enclosing an object.

With some additional transformation rules, the following sentences could readily be added; though they are at present negative instances.

E 21 What is the largest object in Fig. 1.

E 22 When will 2 be just above 3.

E 23 How will 2 get to where 2 is just above 3.

The following illustrate a few sentences outside the content-scope of this algorithm, though they are syntactically of the same type and pertain to the same domain of discourse.

F 1. How many triangles are in Fig. 1.

F 2. Why did 1 move from where 1 was above 2 to where 1 was below 2.

F 3. How can 1 move from where 1 is above 2 to where 1 is below 2.

F 4. Is it true that 1 is large and dark

F 5. How much larger than 1 is 2.

F 6. Is it true that 1 is related to 2 as 3 is related to 4.

F 7. What is a triangle.

One might characterize the typical program represented by the flow diagrams that result from this translation algorithm as that of nested loops involving search routines. Fig. IV is a representative flow diagram. There is relatively little computation, mostly string matching, string replacement, and string definition. Self-modifying programs would not be produced. Yet the range of programs represented by the flow diagrams is quite large, for particular strategies for searching the data base and indexes to it, for making inferences and for parsing most efficiently are not specified.

Moreover, the efficiency of any program depends critically upon how the diagram descriptions are represented for computer storage and used by the program, and how the entire data base is structured. But this, as well as the problem of how to relate portions of incomplete diagrams, require separate investigation. We cannot, therefore, assess the efficiency of the class of programs represented by our flow diagrams.

The evidence in the form of examples of questions the algorithm can process supports the conclusion that the English-like source language of such questions has a potentially wide scope in that it includes many of the fundamental question-types.

REFERENCES

1. Amarel, S., "An Approach to Problem-Solving By Computer," Part II of Final Report AFCRL-62-367, Contract No. AF19(604)-8422 (May 1962).

2. Amarel, S., "An Approach to Heuristic Problem Solving and Theorem-Proving in Propositional Calculus," in Systems and Computer Science, Hart and Takasu, eds., University of Toronto Press, 1967.

3. Amarel, S., "On Representations of Problems of Reasoning About Actions," in Machine Intelligence 3, Michie, ed., Edinburgh University Press, 1968.

4. Bobrow, D.G., "Syntactic Analysis of English By Computer - A Survey," Proc. Fall Joint Computer Conference, 1963, pp. 365-387.

5. Bohnert, H., "An English-Like Extension of an Applied Predicate Calculus," in Final Tech. Report on High-Speed Document Perusal by M. Kochen, AFOSR-2817, May 1, 1962, pp. 83-96.

6. Carnap, R., Foundations of Logic and Mathematics, International Encyclopedia of Unified Science, 1, No. 3, Univ. Chicago Press, Chicago, 1939.

7. Charney, E.K., "On the Semantical Interpretation of Linguistic Entities that Function Structurally," presented at First International Conference on Machine Translation and Applied Language Analysis, September 1961.

8. Chomsky, N., Cartesian Linguistics, Harper and Row, New York, 1966.

9. Cooper, W.S., "Fact Retrieval and Deductive Question-Answering Information Retrieval Systems," J. ACM 11, 1964, pp. 117-137.

10. Craig, J.A. et al., "DEACON: Direct English Access and Control," AFIPS Proc. 29, 1966, p. 365.

11. Craft, J.L., Goldman, E.H., and Strohm, W.B., "A Table Lookup Machine for Processing of Natural Languages," IBM J. of Res. & Dev. 5, No. 3, July 1961, pp. 192-203.

12. Darlington, T.L., "Translating Ordinary Language into Symbolic Logic," MAC-M-149, Cambridge, Mass., M.I.T., March 1964.

13. D'Imperio, Mary, "Data Structures and Their Representation in Storage," Parts I and II, NSA Technical Journal, IX, Nos. 3 & 4, 1964, pp. 59-81, 7-54.

14. Evans, T., "A Heuristic Program to Solve Geometric Analogy Problems," Proc. Spring Joint Computer Conference 25, 1964, pp. 327-338.

15. Feigenbaum, E. and Lederberg, J., DENDRAL, Interim Report to NASA, NSG-81-60, 1966.

16. Green, B.F., Wolf, A.K., Chomsky, C., and Langhery, K., "BASEBALL: An Automatic Question Answerer," Computers and Thought, (E.A. Feigenbaum and J. Feldman, eds.), McGraw-Hill, New York, 1963, pp. 207-216.

17. Irons, E.T., "The Structure and Use of The Syntax-Directed Compiler," Annual Review in Automaton Programming, Pergamon Press 3, 1963, pp. 207-227.

18. Kasher, Asa, Data-Retrieval by Computer: A Critical Survey," The Growth of Knowledge: Selected Readings in the Organization and Retrieval of Information, (M. Kochen, ed.), J. Wiley & Sons, New York, 1967.

19. Katz, J.A. and Fodor, J.A., "The Structure of Semantic Theory," Language 39, 1963, pp. 170-210.

20. Kirsch, R.A., "Computer Interpretation of English Text and Picture Patterns," IEEE Trans. in Electronic Computers, 1954.

21. Kirsch, R.A., Ray, L.C. and Cahn, L., Urban, G.H., "Experiments in Processing Pictorial Information with a Digital Computer," Report No. 5713, N.B.S., December 1957.

22. King, G.W., "Table Look-up Procedures in Language Processing, Part I, The Raw Text," IBM J. of Res. & Dev. 5, 1961, p. 86.

23. Kochen, M., "Adaptive Mechanisms in Digital Concept-Processing," Final Report on Adaptive Man-Machine Concept-Processing to AFCRL, June 14, 1962, pp. 104-156. Reprinted in The Growth of Knowledge: Selected Readings on the Organization and Retrieval of Information, (M. Kochen, ed.), J. Wiley & Sons, New York, 1967.

24. Kochen, M., Translation of English-Like Queries Into Efficient Computer Search Programs for Question Answering, RCA Laboratories Report, November 1965.

25. Kochen, M., "On the Representation of Limited Information by Means of Pictures, Trees, and English-Like Sentences," Scientific Report No. 2, Contract No. AF49(638)-1184, May 1968.

26. Kochen, M., Some Problems in Information Science, Scarecrow Press, New York, 1965.

27. Kondo, M. and Murata, H., "On Proof Retrieval: Problem-Solving Machines, I," Proc. Japan Acad. 41, No. 4, April 1965, pp. 254-259.

28. Lindsay, R.K., "Inferential Memory as the Basis of Machines Which Understand Natural Language," in Computers and Thought, (E.A. Feigenbaum and J. Feldman, eds.), McGraw-Hill, New York, 1963.

29. Londe, D. and Simmons, R.F., "Namer: A Pattern-Recognition System for Generating Sentences About Relations Between Line Drawings," Proc. ACM, 20th Annual Conference, 1965, pp. 162-175.

30. Minsky, M., "Steps Toward Artificial Intelligence," Proc. IEEE, January 1964, pp. 8-48.

31. Mooers, C., "A Mathematical Theory of Language Symbols in Retrieval," International Conference on Scientific Information, Washington, D.C., 1958, Vol. 2, pp. 1327-1364, and "Some Mathematical Fundamentals of the Use of Symbols in Information Retrieval," Proc. International Conf. on Information Processing, UNESCO, Paris, 1959.

32. Narasimhan, R., "A Linguistic Approach to Pattern Recognition," Univ. Illinois Dig. Comp. Lab. Dept. 121, July 10, 1962, and "Syntax Directed Interpretation of Classes of Pictures," presented at the ACM Workshop on Programming Languages, August 1965.

33. Quillian, R., "Word Concepts: A Theory and Simulation of Some Basic Semantic Capabilities," CIP working paper 79, Carnegie Inst. of Technology, 1965; also: "Semantic Memory," unpublished doctoral dissertation, Carnegie Inst. of Technology, 1966, plus comments to possibly appear in proceedings of AFOSR-sponsored conference on Intelligence, at Athens, Georgia, January 1967.

34. Rankin, B.K., "A Programmable Grammar for a Fragment of English for Use in an Information Retrieval System," N.B.S. Report 7352, June 1961.

35. Simon, H., "Experiments with a Heuristic Computer," Journal of the ACM 10, No. 4, October 1963, pp. 493-506.

36. Simmons, R., "Answering English Questions by Computer: A Survey," Commun. ACM 8, No. 1, January 1965, pp. 53-70; also in The Growth of Knowledge: Selected Readings in the Organization and Retrieval of Information, (M. Kochen, ed.), J. Wiley & Sons, New York, 1967.

37. Thompson, F., "English for the Computer," AFIPS Conf. Proc., Fall Joint Computer Conference, Spartan Books, Washington, D.C., 1966, pp. 343-356.

38. Turing, A.M., "On Computable Numbers with Applications to the Entscheidungs Problem," Proc. London Math. Soc., Ler 2, 42, pp. 230-265, 1936-7; Correction, ibid., 43, pp. 544-546, 1937.

39. van Dam, A., "A Compact Data Structure for Storing, Retrieving and Manipulating Line Drawings," Spring Joint Comp. Conf. 1967.

40. Watt, W.C. and Hsu, R.W., "Concordance to the Rules of PLACEBO V," U.S. Government Res. and Dev. Report 41, 521(A). March 25, 1966, pp. 163-177.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R&D
(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author)
RCA Laboratories
Princeton, New Jersey 08540

2a REPORT SECURITY CLASSIFICATION
Unclassified

2b GROUP
N/A

3 REPORT TITLE
AUTOMATIC QUESTION-ANSWERING OF ENGLISH-LIKE QUESTIONS ABOUT SIMPLE DIAGRAMS

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)
Scientific Report No. 3

5 AUTHOR(S) (Last name, first name, initial)
Kochen, Manfred

6 REPORT DATE	7a TOTAL NO. OF PAGES	7b NO. OF REFS
May 1968	55	40

8a CONTRACT OR GRANT NO	9a ORIGINATOR'S REPORT NUMBER(S)
AF 49(638)- 1184	
b PROJECT NO	
c	9b OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d	

10 AVAILABILITY/LIMITATION NOTICES
Qualified users may request copies of this report from DDC.

11 SUPPLEMENTARY NOTES	12 SPONSORING MILITARY ACTIVITY
	Air Force Office of Scientific Research Office of Aerospace Research Arlington, Va. 22209

13 ABSTRACT

This paper presents a technique for translating certain English-like questions into procedures for answering them in order to explore how large a class of basic question types can be so processed. The English-like questions all pertain to simple diagrams built of elementary figures with relations like "above" and "larger than." The input to the program into which the algorithm presented here could be implemented are questions such as "Is it true that in Fig. 1 each triangle is above a circle," and may include terms like "how," "when," "what" in an interesting variety of interrogative sentence types. The output of the program is a flow diagram for another program to answer the question by inference and search of a structured data base in which representations of diagrams are stored. The English-like source language of questions that the algorithm can process, though restricted and fixed in syntax and domain of discourse, has a potentially wide scope in that it includes some of the fundamental question types.

DD FORM 1 JAN 64 1473

UNCLASSIFIED
Security Classification

UNCLASSIFIED
Security Classification

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Artificial Intelligence Question-Answering Graphic Data Processing Syntax-Directed Compilers Picture Analysis by Computer Inference by Computer Language Processing Flow-Diagram Compilation Problem-Solving Machines						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

UNCLASSIFIED
Security Classification